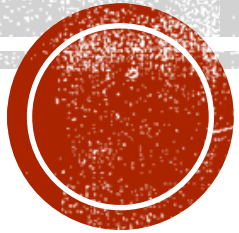
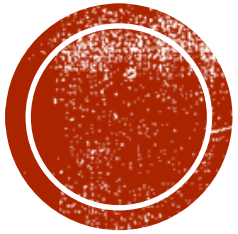
THERE IS NO SILVER BULLET

Alberto Mori

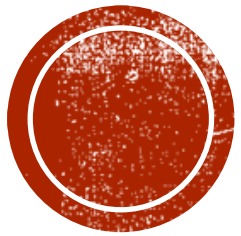
Software Developer @ Able Tech srl





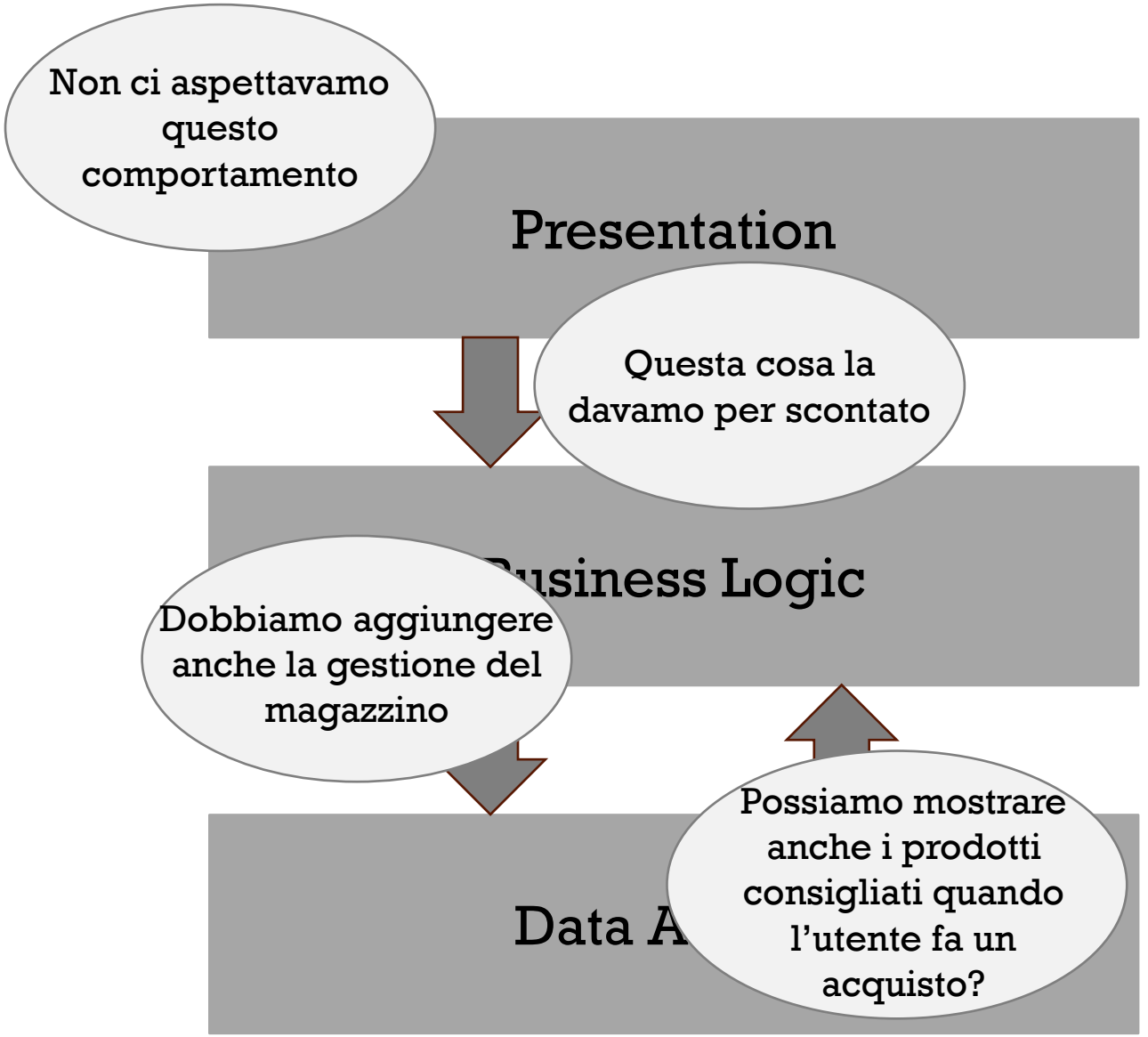
IL TITOLO È UNA CITAZIONE INVOLONTARIA AL PAPER «NO SILVER BULLET – ESSENCE AND ACCIDENT IN SOFTWARE ENGINEERING» DI FRED BROOKS

https://www.cs.unc.edu/techreports/86-020.pdf



NESSUN PATTERN È STATO MALTRATTATO DURANTE LA PREPARAZIONE DEL TALK

«La mia applicazione è una cosa semplice, deve solo leggere e scrivere da un database»



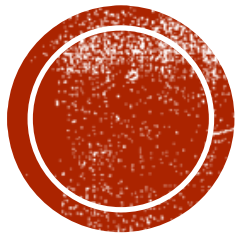

PAURA, EH?

MY SILVER BULLETS

Ubiquitous Language

Bounded Context

CQS
CQRS



UBIQUITOUS LANGUAGE

UBIQUITOUS LANGUAGE…CHI?!

«A common, rigorous language between developers and users»

https://martinfowler.com/bliki/UbiquitousLanguage.html

CHE VANTAGGI PORTA?

~~Elimina~~ Assottiglia la frizione tra tutti gli stakeholder

Facilita la comprensione del dominio del problema che dobbiamo risolvere

Velocizza nel momento in cui devo risolvere problematiche che emergono durante lo sviluppo

DIFFICOLTÀ

Ego tecnico

Spesso è difficile tenere sotto controllo l'ambiguità

Gli stakelhoder devono guidarsi a vicenda

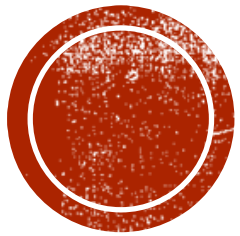
The Five Orders of Ignorance - Phillip G. Armour (github.com)

COME POSSO APPLICARLO?

Workshop o sessioni di esplorazione del dominio (es. Event Storming)

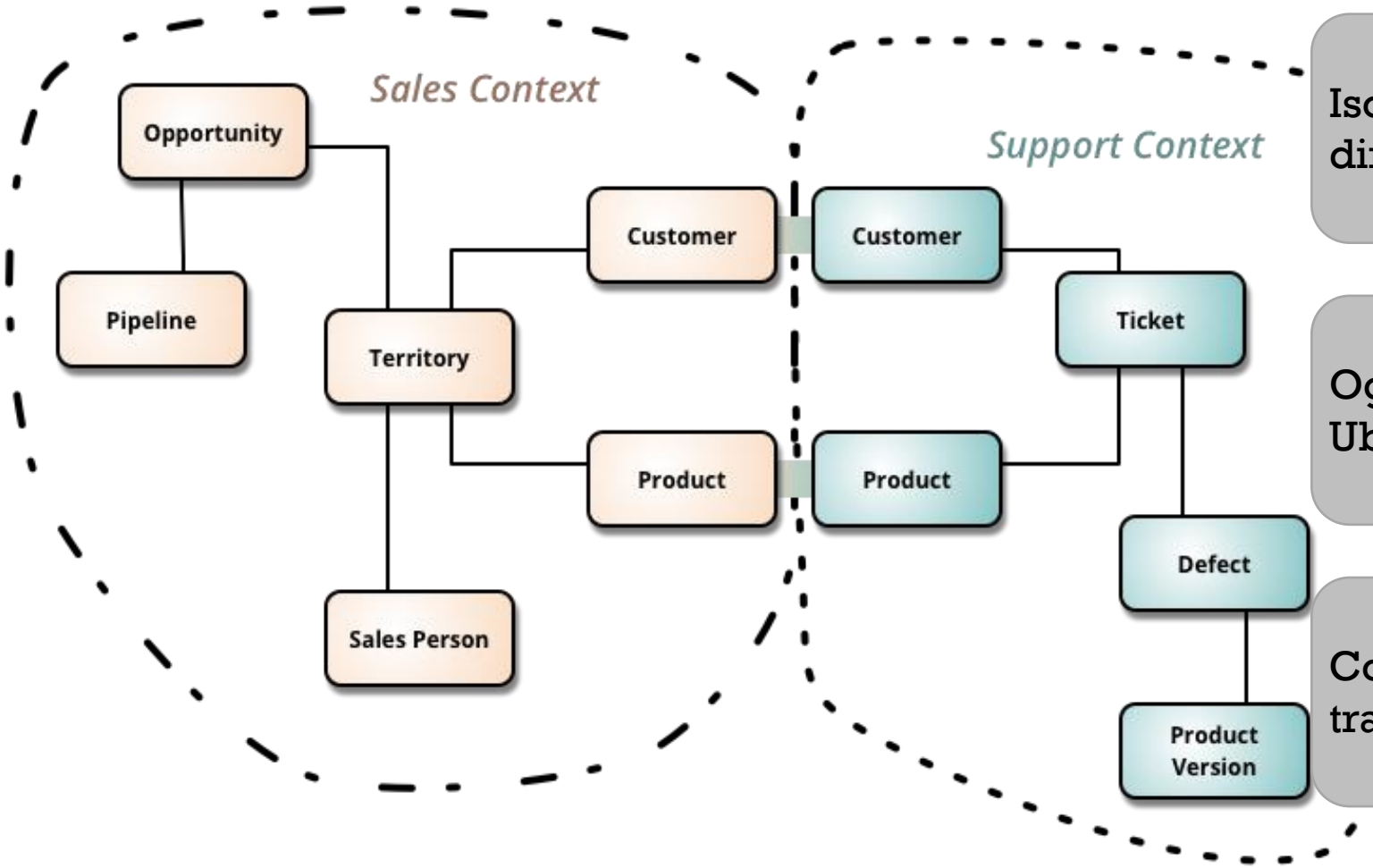
Mantenere il più possibile l'ubiquitous language nella definizione dei task / backlog items / user stories

Usare i test come aiuto nella definizione della funzionalità



BOUNDED CONTEXT

COSA È UN BOUNDED CONTEXT?



Isolati e senza riferimenti diretti tra di loro

Ognuno ha il proprio Ubiquitous Language

Comunicano con altri BC tramite Context Mapping

CONTEXT MAPPING

Strategie che definiscono come Bounded Context diversi comunicano tra di loro

Customer Supplier

Mutually dependent

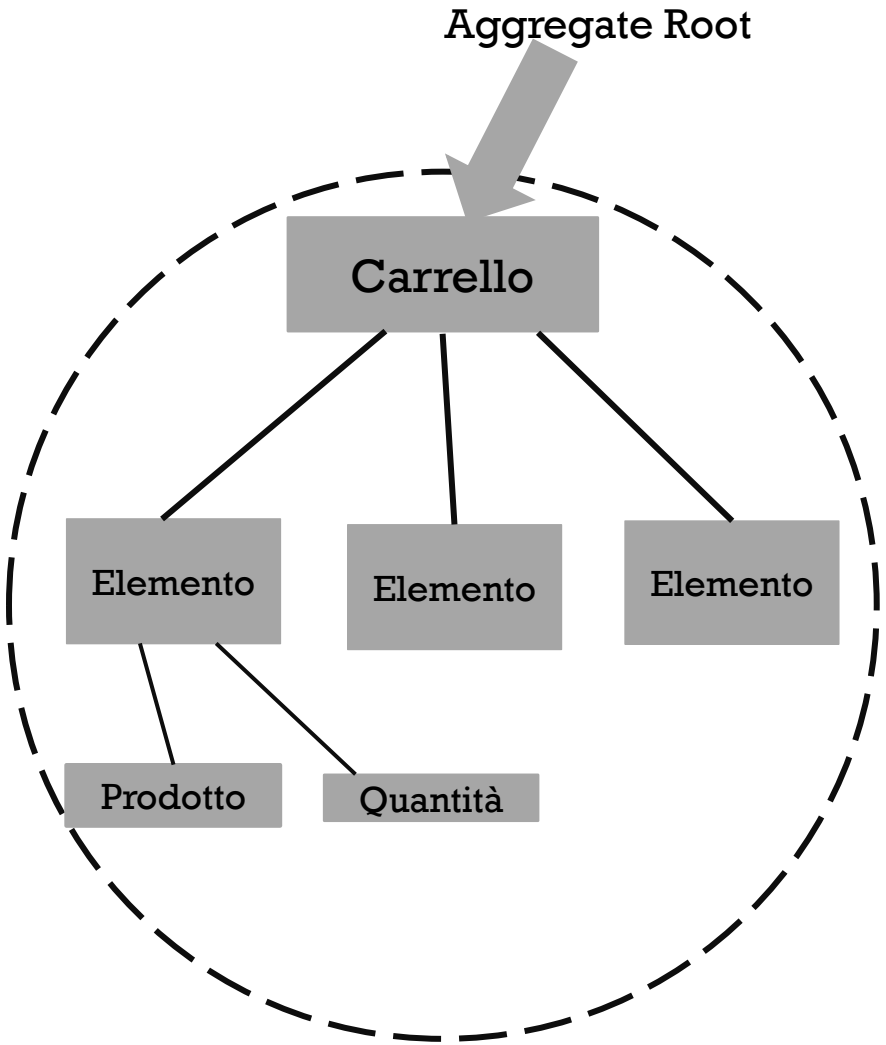
Free

AGGREGATE

Grafo di oggetti collegati tra di loro secondo la logica di dominio

Espone *«both data and behavior»*

La gestione dell'aggregate passa attraverso un singolo attore chiamato **Aggregate Root**



QUALI SONO I VANTAGGI?

Separazione delle responsabilità

Maggiore autonomia nello sviluppo da parte di team diversi

Minore impatto delle modifiche fatte a parti che non interessano direttamente il nostro Bounded Context

TUTTO MOLTO BELLO MA...

L' individuazione dei Bounded Context e degli aggregati non è sempre facile

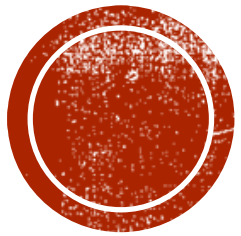
Evitare le dipendenze dirette tra Bounded Context potrebbe non essere così immediato

COME POSSO IMPLEMENTARLO?

Una buona esplorazione del dominio è fondamentale

Separare ogni Bounded Context in un progetto diverso

Nel progetto di Context mapping esponete solo l'interfaccia di comunicazione e non l'effettiva implementazione



CQS / CQRS

CQS VS CQRS

CQS: ogni unità di codice o restituisce dei dati oppure esegue un'azione che modifica lo stato del sistema. Chi legge non può modificare nulla, chi scrive non ritorna dati.

CQRS: ogni attore presente nel mio sistema deve sapere fare solo una cosa tra le due. O restituisce dei dati oppure è in grado di modificare lo stato.

PROS & CONS

Separazione delle responsabilità

Manutenzione più facile

Testabilità maggiore

Il codice da scrivere aumenta

La complessità (soprattutto nel design) potrebbe aumentare

Non è così semplice scrivere i test

COME LO IMPLEMENTO?

Lato Query posso implementare ogni use case come un metodo di un servizio. Questo servizio potrebbe comunicare con lo strato di persistenza nascondendo la capacità di modifica degli oggetti

Lato Command posso utilizzare pattern come Transaction Script oppure posso modellare ogni comando come un singolo messaggio (se ho un sistema basato su messaggistica)

Ocio all'Ubiquitous Language!

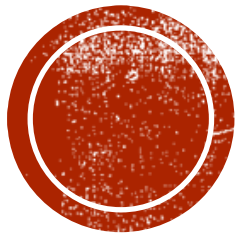
HOW I DID IT

CONCLUDENDO

NON esistono silver bullet!

Alcuni approcci, secondo me, possono aiutare a tenere sotto controllo la complessità

Context is king



GRAZIE!

RESTIAMO IN CONTATTO

https://www.linkedin.com/in/morialberto

@albx

@albx87

@albx87

https://t.me/+_kyyKaympJ5kOWQ0

https://www.morialberto.it

UGIdotNET - Spike Time

FEEDBACK